

## **Agile Development With PHP**

Conférence PHP Québec 2004  
Montreal, Canada  
March 25, 2004

Alex Pukinskis  
alex@satoridesign.com  
<http://satoridesign.com>

# Why Agile?

Software development is hard. Many software projects end up

- Late
- Cancelled
- Over budget
- Full of unmaintainable code
- Delivered but unusable due to changing requirements
- Never used by the end users

# **Developing software without any process**

- is common in the PHP world
- works on very small projects
- allows you to go fast initially
- larger projects suffer from entropy

# What is Entropy?

- Disorder or randomness
- In software, “brittle” code
- Fix one bug and another one pops up
- Results from inadequate or incorrect attention to design

# Traditional Software Development methodologies (i.e. Waterfall)

- Aim to manage entropy by controlling the process
- Break development into phases: Requirements, Architecture, Coding, Testing, Deployment, Maintenance
- Once a phase is over, that part of the project is finished.
- Works as a “defined process control” system - if you control the inputs and process, you should get predictable output

# Failures of Defined Process Control in Software Development

- Requirements gathering is never perfect - new requirements surface when users use the system
- Architecture can't be "locked down" because of changing requirements.
- Entropy increases because original design isn't appropriate for the needs of the system - can cause stalled projects or unfixable bugs
- Cumbersome change management processes make it difficult for the customer to change their mind about what they wanted

# Process Control in Agile Development

- Empirical process control
- Take aim towards a goal, measure success, and correct as new goals are defined
- Recognizes that requirements change
- Adaptive rather than Predictive

# Agile Development

Agile Alliance  
<http://agilealliance.org>

“To satisfy the customer through early and continuous delivery of valuable software.”

# Agile Manifesto

Individuals and Interactions over Processes and Tools  
Working Software over Comprehensive Documentation  
Customer Collaboration over Contract Negotiation  
Responding to Change over Following a Plan

While there is value in the items on the right,  
we value the items on the left more.

# Extreme Programming

*Separating business and technical decisions*

4 key values:

- Communication
- Simplicity
- Feedback
- Courage

# Extreme Programming Practices

- The Planning Game
- Testing
- Pair Programming
- Refactoring
- Simple Design
- Collective Code Ownership
- Continuous Integration
- On-Site Customer
- Small Releases
- Sustainable Pace
- Coding Standards
- System Metaphor / Common Language / Domain Model

The practices work together

## Agile Tools for PHP

- Mostly consist of testing tools
- Many versions of PHPUnit exist
- PHPUnit from PEAR is most popular (Sebastian Bergemann)
- I use a modified PHPUnit by Vincent Oostindie (<http://www.students.cs.uu.nl/people/voostind/index.php?page=software>) that adds sounds for the “Pavlovian” effect.
- Easy to build your own - you just need a few types of assertions, a test runner, and a results display.

# PHP and Acceptance Testing

- Verify that the system meets requirements
- Easiest acceptance testing is by hand, if your app is small enough
- Easy to automate acceptance testing with PHPunit, since you can set up `$_SESSION`, `$_GET`, `$_POST`, etc.
- Downside of using PHPunit is that programmers need to write the tests; it's often better for a non-programmer to do the testing.
- Could build your own tool for testing business logic, like <http://fit.c2.com/>

## **PHP/Agile Challenges**

- Agile works best with Object-oriented development; weak OO support in PHP 4 makes this more difficult
- This should improve with PHP 5
- Using modules like PEAR DB seems to contradict YAGNI (You Aren't Going to Need It) and simple design.
- Lack of an architectural standard makes it easy to create an architectural mess, which is especially difficult to recover from if you don't do test-driven development

# 5 things your team can do to become more agile

- The Planning Game
- Daily Standup Meetings
- Coding Standards
- Continuous Integration
- Small Releases

Other XP practices (testing, pairing, refactoring, simple design) will give you a much bigger payoff, but these are the easy ones to do right away

# The Planning Game

- Get your requirements into user stories that the business people can understand
- Stories should not be technical (i.e. “Implement DB Connection Pool” is a bad story)
- Developers estimate - do this even if your estimates are ignored!
- Estimated time  $\neq$  real time. Measure your velocity (Estimated units per iteration)
- Business people prioritize stories

# Daily Standup Meetings

- Each person says only what they did yesterday, what they are doing today, and what's standing in their way.
- Managers should work at removing impediments
- Keep it short! Design discussions happen after the meeting.

# Coding Standards

- Stop arguing - pick something!
- It doesn't matter which standard you pick - just be consistent
- Agree on variable naming conventions
- Start making sure your code expresses its intent clearly (a person who understands the programming language and problem domain should easily be able to tell how your code works).

# Continuous Integration

- Use source control - CVS is free and easy and there's no excuse not to.
- Check in changes as often as possible - multiple times each day
- The code in source control should be bug-free, should always run without errors, and the tests should pass.
- Try to ensure that everyone is working from the same code base.

# Small Releases

- Produce business value every 1-4 weeks
- Each release is fully tested, bug-free, and ready to go
- Business people decide whether it goes to the end users.

## **Resources**

*XP Mailing List*

<http://groups.yahoo.com/group/extremeprogramming/>

*Agile Alliance*

<http://agilealliance.org>

*XP Overview*

<http://extremeprogramming.org/>

*Unit Testing with PHP*

<http://www.phppatterns.com/index.php/article/articleprint/33/-1/2/>

## **Books**

Extreme Programming Applied (Ken Auer, Roy Miller)

User Stories Applied For Agile Software Development (Mike Cohn)